APPENDIX

A RELATED WORKS

Deep Tabular Learning The tabular domain, distinct from image and language data, has seen limited exploration in deep learning, primarily due to the absence of spatial and semantic relationships that convolutional or recurrent neural networks rely on. Although attention-based architectures like TabNet (Arik & Pfister, 2021), FT-Transformer (Gorishniy et al., 2021) have been introduced for tabular data, they often require large datasets, resulting in computational overhead that surpasses performance gains. As a result, multi-layer perceptrons (MLPs) remain dominant in practical applications.

Test-Time Adaptation In recent studies, a novel approach called test-time adaptation (TTA) has emerged as a solution to address limitations found in traditional unsupervised domain adaptation (UDA) methods (Ganin & Lempitsky, 2015; Wang et al., 2021b). Test-time adaptation strategies (Wang et al., 2021a; Liu et al., 2021; Niu et al., 2023; Zhou et al., 2023), aim to adapt the pre-trained model from a source domain to a target domain, without access to source data. While these TTA methods have shown promising performance, the test time adaptation scheme for the tabular domain still remains under-explored regime since the tabular domain poses unique challenges due to its own nature. Therefore, there is a strong motivation to explore and develop TTA methods tailored to the tabular realm.

Uncertainty Calibration Uncertainty calibration is a crucial technique for enhancing the reliability of deep learning model outputs. It involves estimating the model's confidence in its predictions by examining the probabilities assigned to predicted classes. Conventional training methods often result in unwarranted overconfidence in model outputs, prompting research into uncertainty calibration methods, such as Platt scaling (Platt, 2000) and isotonic regression (Stylianou & Flournoy, 2002) predate the era of deep learning, while more recent methods, including beta calibration (Kull et al., 2017) and Dirichlet calibration (Kull et al., 2019) rooted in probability distributions, have emerged. Temperature scaling, akin to Platt scaling but simpler, directly impacts uncertainty while keeping model predictions intact.

Label Distribution Shift In the domain adaptation area, the assumption that there is no label distribution shift can be risky, as such shifts can occur readily and significantly affect model performance (Quinonero-Candela et al., 2008). Various approaches have been developed to tackle this issue including ReMixMatch (Berthelot et al., 2020), online label adaptation (Wu et al., 2021), black box shift estimation (BBSE) (Lipton et al., 2018). Recently, in the computer vision domain, some works (Park et al., 2023; Zhou et al., 2023) have been proposed to consider label distribution shift under the test-time adaptation setting. In our work, we leverage the model's predictions and training set statistics to adjust the output. Additionally, we integrate a label adapter to effectuate substantive modifications to the output.

B Additional Observations

B.1 Entropy Distributions

To show the generalizability of the observation in Section 2.1, wherein prediction entropy of the model consistently exhibits a strong bias toward the under-confident region, we additionally provide entropy distribution histograms for test instances across six different datasets and three representative deep tabular learning architectures. Here, we can observe distinct patterns between the upper four rows (HELOC, ANES, Diabetes Readmission, CMC) and the lower two rows (MFEAT-PIXEL, DNA). In the upper four rows, the entropy is consistently high, indicating a skew towards the under-confident region. However, in the lower two rows, this is not the case. This discrepancy arises from the fact that the two datasets below are characterized by homogeneity in all columns, one being a linearized image dataset (MFEAT-PIXEL) and the other a DNA string sequence dataset (DNA). This comprehensive analysis showcases the unique characteristics of tabular data – biased entropy distributions toward the under-confident region – compared to other domains. As discussed in Section 2.1, applying entropy minimization with samples of high entropy often cause gradient exploding and model collapse (Niu et al., 2023).

B.2 Latent Space Visualizations

We further visualize latent spaces for test instances using t-SNE across six different datasets and three representative deep tabular learning architectures, to show the common behavior of the observation in Section 2.1, wherein tabular data exhibits extremely complex decision boundary within the latent space compared to that of other domains. Again, by comparing upper for rows with tabular dataset and lower two rows with linearized image dataset (MFEAT-PIXEL) and homogeneous DNA string sequence dataset (DNA), it is obvious that the decision boundary within the latent space of tabular domain is particularly complex compared to other domains except for the case of DNA dataset with TabNet model. As shown in Section 2.1, this also provides another indication of the limitations of existing TTA methods (Sun et al., 2020; Gandelsman et al., 2022; Liu et al., 2021; Gandelsman et al., 2022; Boudiaf et al., 2022; Zhou et al., 2023), which heavily rely on the cluster assumption.

Algorithm 1 AdapTable

```
1: Input: Pre-trained tabular classifier on the source domain $\mathcal{F}(\cdot|\theta) : \mathbb{R}^D \to \mathbb{R}^C$, post-trained shift-aware uncertainty calibrator $\mathcal{G}(\cdot, \cdot|\phi)$, indicator function $\mathbb{1}$, quantile function $Q$, Softmax and $L_1$ normalization functions $\sigma_{\text{Softmax}}(\cdot)$, $\sigma_{L_1}(\cdot)$, tabular data in the source domain $\mathcal{D}_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_i$, current tabular batch in the target domain $\{\boldsymbol{x}_i^t\}_{i=1}^N$
2: Parameters: Smoothing factor $\alpha$, Low/high quantiles $q_{\text{low}}/q_{\text{high}}$
3: $\boldsymbol{p}_s(y),\ T \leftarrow \big(\sum_{i=1}^{|\mathcal{D}_s|} \mathbb{1}_{[j=y_i^s]}/|\mathcal{D}_s|\big)_{j=1}^C,\ 3\max_j \boldsymbol{p}_s(y)_j/2\min_j \boldsymbol{p}_s(y)_j$
4: if $\{\boldsymbol{x}_i^t\}_{i=1}^N$ is the first test batch then
5:     $\boldsymbol{p}_t^{\text{oe}}(y) \leftarrow \big(1/C\big)_{j=1}^C$    ▷ Initialize online target label estimator
6: end if
7: for $u = 1$ to $D$ do
8:     $\boldsymbol{s}_u^t \leftarrow \big(\boldsymbol{x}_{iu}^t - (\sum_{i'=1}^{|\mathcal{D}_s|} \boldsymbol{x}_{i'}^s/|\mathcal{D}_s|)_u\big)_{i=1}^N$    ▷ Calculate shift information of $u$-th column
9: end for
10: for $i = 1$ to $N$ do
11:     $\boldsymbol{p}_t(y|\boldsymbol{x}_i^t) \leftarrow \sigma_{\text{Softmax}}\big(\mathcal{F}(\boldsymbol{x}_i^t|\theta)\big)$
12:     $\boldsymbol{p}_t^{\text{de}}(y|\boldsymbol{x}_i^t) \leftarrow \sigma_{L_1}\big(\boldsymbol{p}_t(y|\boldsymbol{x}_i^t)/\boldsymbol{p}_s(y)\big)$    ▷ Predict debiased target label estimator
13:     $T_i \leftarrow \mathcal{G}\big(\mathcal{F}(\boldsymbol{x}_i^t|\theta), \boldsymbol{s}^t|\phi\big)$    ▷ Determine per-sample temperature of $\boldsymbol{x}_i^t$
14:     $j^*,\ j^{**} \leftarrow \arg\max_{1\le j\le C} \boldsymbol{p}_t(y|\boldsymbol{x}_i^t)_j,\ \arg\max_{1\le j\le C, j\ne j^*} \boldsymbol{p}_t(y|\boldsymbol{x}_i^t)_j$
15:     $\epsilon_i \leftarrow 1/\big(\sigma_{\text{Softmax}}\big(\mathcal{F}(\boldsymbol{x}_i^t|\theta)/T_i\big)_{j^*} - \sigma_{\text{Softmax}}\big(\mathcal{F}(\boldsymbol{x}_i^t|\theta)/T_i\big)_{j^{**}}\big)$    ▷ Define uncertainty of $\boldsymbol{x}_i^t$
16: end for
17: $\boldsymbol{p}_t(y) = (1-\alpha)\cdot\sum_{i=1}^N \boldsymbol{p}_t^{\text{de}}(y|\boldsymbol{x}_i^t)/N + \alpha\cdot\boldsymbol{p}_t^{\text{oe}}(y)$    ▷ Estimate current target label distribution
18: for $i = 1$ to $N$ do
19:     if $\epsilon_i \le Q\big(\{\epsilon_{i'}\}_{i'=1}^N, q_{\text{low}}\big)$ then
20:         $c_i \leftarrow T$
21:     else if $Q\big(\{\epsilon_{i'}\}_{i'=1}^N, q_{\text{low}}\big) < \epsilon_i < Q\big(\{\epsilon_{i'}\}_{i'=1}^N, q_{\text{high}}\big)$ then
22:         $c_i \leftarrow 1$    ▷ Measure temperature $c_i$ using uncertainty $\epsilon_i$
23:     else
24:         $c_i \leftarrow 1/T$
25:     end if
26:     $\boldsymbol{p}_t(y|\boldsymbol{x}_i^t)' \leftarrow \sigma_{\text{Softmax}}\big(c_i\cdot\mathcal{F}(\boldsymbol{x}_i^t|\theta)\big)$    ▷ Adjust original probability with $c_i$
27:     $\hat{\boldsymbol{p}}_i(y) \leftarrow \boldsymbol{p}_t(y|\boldsymbol{x}_i^t)'/2 + \sigma_{L_1}\big(\boldsymbol{p}_t(y|\boldsymbol{x}_i^t)'\boldsymbol{p}_t(y)/\boldsymbol{p}_s(y)\big)/2$    ▷ Perform self-ensembling
28: end for
29: $\boldsymbol{p}_t^{\text{oe}}(y) \leftarrow (1-\alpha)\cdot\sum_{i=1}^N \hat{\boldsymbol{p}}_i(y)/N + \alpha\cdot\boldsymbol{p}_t^{\text{oe}}(y)$    ▷ Update online target label estimator
30: Output: Final predictions of $\{\hat{\boldsymbol{p}}_i(y)\}_{i=1}^N$
```

B.3 Reliability Diagrams

We also provide further reliability diagrams across six different datasets and three representative deep tabular learning architectures, to show that tabular data often exhibits both overconfident and under-confident patterns compared to consistent overconfident behavior in image domain (Stylianou & Flournoy, 2002), and under-confident behavior in graph domain (Wang et al., 2021c). In cases of overconfident confidence is evident, whereas in HELOC, (CMC, TabNet), and (DNA, TabNet), under-confident confidence is observed. This underscores the necessity for the proposition of uncertainty calibrators tailored specifically to the tabular domain.

C Detailed Algorithm of AdapTable

The overall procedure of the proposed *AdapTable* is minutely summarized in Algorithm 1. Given a pre-trained tabular classifier $\mathcal{F}(\cdot|\theta) : \mathbb{R}^D \to \mathbb{R}^C$ on the source domain $\mathcal{D}_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_i$, and current tabular batch in the target domain $\{\boldsymbol{x}_i^t\}_{i=1}^N$, we first post-train shift-aware uncertainty calibrator $\mathcal{G}(\cdot, \cdot|\phi)$ by optimizing ϕ using training set again after training $\mathcal{F}$ with loss function

$$\mathcal{L} = \mathcal{L}_{\text{FL}} + \lambda_{\text{CAL}}\mathcal{L}_{\text{CAL}}.$$

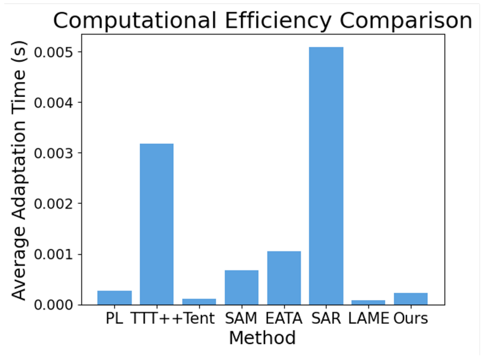

Figure 6: Computational efficiency comparison between test-time adaptation baselines and AdapTable. We report the average adaptation time calculated over all test instances in CMC dataset corrupted by Gaussian noise.

Here, we calculate shift information of u-th column for $1 \le u \le D$ as

$$\boldsymbol{s}_u^s = \big(\boldsymbol{x}_{iu}^s - (\sum_{i'=1}^{|\mathcal{D}_s|} \boldsymbol{x}_{i'}^s / |\mathcal{D}_s|)_u\big)_{i=1}^N,$$

with $\lambda_{\text{CAL}} = 0.1$ denotes the weight of regularization loss. Given the original probability $\boldsymbol{p}_s(y|\boldsymbol{x}_i^s) = \sigma_{\text{Softmax}}\big(\mathcal{F}(\boldsymbol{x}_i^s|\theta)\big)$, and calibrated prediction probability $\boldsymbol{p}_i = \sigma_{\text{Softmax}}\big(\mathcal{F}(\boldsymbol{x}_i^s|\theta)/T_i\big)$ with per-sample tempperature $T_i = \mathcal{G}\big(\mathcal{F}(\boldsymbol{x}_i^s|\theta), \boldsymbol{s}^s|\phi\big)$, $j^* = \arg\max_{1\le j\le C} \boldsymbol{p}_{ij}$, and $j^{**} = \arg\max_{1\le j\le C, j\ne j^*} \boldsymbol{p}_{ij}$, the focal loss $\mathcal{L}_{\text{FL}}$ (Lin et al., 2017a) and the regularization loss $\mathcal{L}_{\text{CAL}}$ (Wang et al., 2021c) are defined as follows:

$$\mathcal{L}_{\text{FL}}(\boldsymbol{x}_i^s, y_i^s) = \sum_{j=1}^{C} \mathbb{1}_{[j=y_i^s]}\big(1-\boldsymbol{p}_{ij}\big)^\gamma \log\big(\boldsymbol{p}_{ij}\big),$$

$$\mathcal{L}_{\text{CAL}}(\boldsymbol{x}_i^s, y_i^s) = \mathbb{1}_{[j^*=y_i^s]}\big(1-\boldsymbol{p}_{ij^*}+\boldsymbol{p}_{ij^{**}}\big) + \mathbb{1}_{[j^*\ne y_i^s]}\big(\boldsymbol{p}_{ij^*}-\boldsymbol{p}_{ij^{**}}\big).$$

Here, $\mathcal{L}_{\text{CAL}}$ alienates $\boldsymbol{p}_{ij^*}$ and $\boldsymbol{p}_{ij^{**}}$ for correctly predicted samples whereas it attracts them for the other ones, and $\gamma = 2$. For each epoch, we apply cosine annealing scheduler.

After post-training $\mathcal{G}$, we calculate shift information of u-th column for $1 \le u \le D$ as $\boldsymbol{s}_u^t = \big(\boldsymbol{x}_{iu}^t - (\sum_{i'=1}^{|\mathcal{D}_s|} \boldsymbol{x}_{i'}^s/|\mathcal{D}_s|)_u\big)_{i=1}^N$. Then, we predict per-sample temperature of $\boldsymbol{x}_i^t$ as $T_i = \mathcal{G}\big(\mathcal{F}(\boldsymbol{x}_i^t|\theta), \boldsymbol{s}^t|\phi\big)$ and define uncertainty ϵ_i of $\boldsymbol{x}_i^t$ as the reciprocal of the margin of the calibrated probability like below:

$$\epsilon_i = 1/\big(\sigma_{\text{Softmax}}\big(\mathcal{F}(\boldsymbol{x}_i^t|\theta)/T_i\big)_{j^*} - \sigma_{\text{Softmax}}\big(\mathcal{F}(\boldsymbol{x}_i^t|\theta)/T_i\big)_{j^{**}}\big).$$

With previously calculated online target label estimator, we predict the debiased target label estimator $\boldsymbol{p}_t^{\text{de}}(y|\boldsymbol{x}_i^t)$ and estimate the current target label distribution $\boldsymbol{p}_t(y)$ with

$$\boldsymbol{p}_t^{\text{de}}(y|\boldsymbol{x}_i^t) = \sigma_{L_1}\Big(\boldsymbol{p}_t(y|\boldsymbol{x}_i^t)/\boldsymbol{p}_s(y)\Big)$$

$$\boldsymbol{p}_t(y) = (1-\alpha)\cdot\sum_{i=1}^{N}\boldsymbol{p}_t^{\text{de}}(y|\boldsymbol{x}_i^t)/N + \alpha\cdot\boldsymbol{p}_t^{\text{oe}}(y).$$

After that, we quantile relative uncertainty ϵ_i among $\{\epsilon_{i'}\}_{i'=1}^N$ within current batch, and perform temperature sharpening for certain samples, whereas we perform temperature smoothing for uncertain samples with $\boldsymbol{p}_t(y|\boldsymbol{x}_i^t)' = \sigma_{\text{Softmax}}\big(c_i\cdot\mathcal{F}(\boldsymbol{x}_i^t|\theta)\big)$, where c_i is defined as in Equation 1. Using Bayes' theorem, we adjust the predicted probability of each instance $\boldsymbol{x}_i$ as $\sigma_{L_1}\big(\boldsymbol{p}_t(y|\boldsymbol{x}_i^t)'\boldsymbol{p}_t(y)/\boldsymbol{p}_s(y)\big)$, and using self-ensembling (Gao et al., 2023), we get the final prediction of $\hat{\boldsymbol{p}}_i(y) = \boldsymbol{p}_t(y|\boldsymbol{x}_i^t)'/2 + \sigma_{L_1}\big(\boldsymbol{p}_t(y|\boldsymbol{x}_i^t)'\boldsymbol{p}_t(y)/\boldsymbol{p}_s(y)\big)/2$, and update online target label estimator as follows:

$$\hat{\boldsymbol{p}}_i(y) = \boldsymbol{p}_t(y|\boldsymbol{x}_i^t)'/2 + \sigma_{L_1}\big(\boldsymbol{p}_t(y|\boldsymbol{x}_i^t)'\boldsymbol{p}_t(y)/\boldsymbol{p}_s(y)\big)/2$$

$$\boldsymbol{p}_t^{oe} \leftarrow (1-\alpha)\cdot\sum_{i=1}^{N}\hat{\boldsymbol{p}}_i(y)/N + \alpha\cdot\boldsymbol{p}_t^{oe}.$$

Table 5: Post-training time analysis of uncertainty calibrator in AdapTable under different scales, encompassing small-scale (MLP + CMC), medium-scale (FT-Transformer + HELOC), and large-scale (TabNet + DIABETES READMISSION).

Setting	Uncertainty Calibrator Training Time (s)
CMC, MLP	4.46
HELOC, FT-Transformer	9.27
DIABETES READMISSION, TabNet	281.16

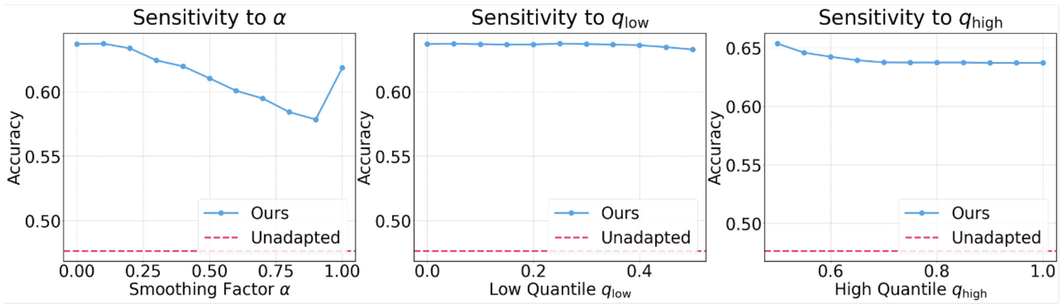

Figure 7: Hyperparameter sensitivity analysis of the proposed AdapTable using MLP under HELOC dataset with respect to smoothing factor α, low uncertainty quantile q_{low}, and high uncertainty quantile q_{high}.

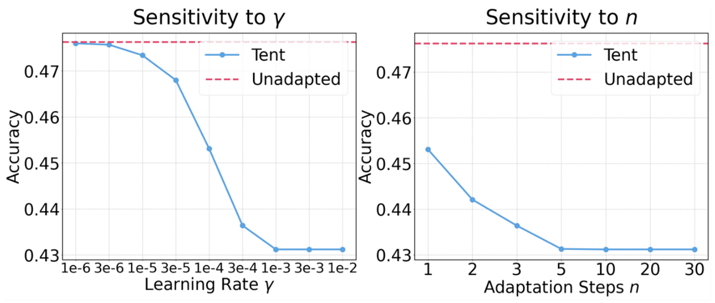

Figure 8: Hyperparameter sensitivity analysis of TENT (Wang et al., 2021a) using MLP under HELOC dataset with respect to learning rate γ, number of adaptation steps n.

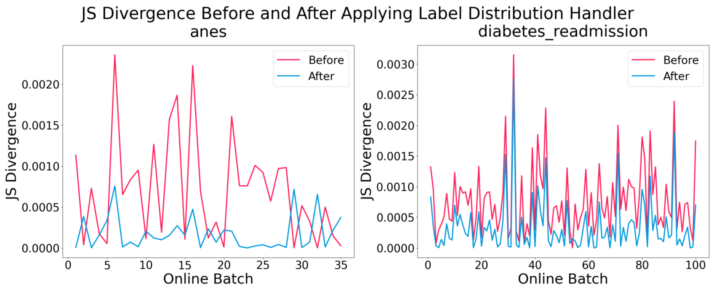

Figure 9: Jensen-Shannon Divergence between estimated target label distribution before and after applying label distribution handler upon ANES and DIABETES READMISSION dataset.

D FURTHER EXPERIMENTS

D.1 COMPUTATIONAL EFFICIENCY

In order to show the efficiency of the proposed AdapTable, we perform a computational efficiency comparison between test-time adaptation baselines and AdapTable in Figure 6. The averaged adapta-

Table 6: Number of total columns, number of numerical and categorical columns along with their classes per datasets used.

Property	HELOC	ANES	DIABETES READMISSION	CMC	MFEAT-PIXEL	DNA
# Columns	22	54	46	9	240	180
# Numerical	20	8	12	2	240	0
# Categorical	2	46	34	7	0	180
# Classes	2	2	2	3	10	3

tion time is calculated by averaging adaptation time over all test instances in CMC dataset corrupted by Gaussian noise. We find that the adaptation time of AdapTable ranks third among eight TTA methods, by showcasing its computational tractability. Furthermore, we observe that our approach requires significantly less adaptation time compared to TTA baselines such as TTT++ (Liu et al., 2021), SAM (Foret et al., 2021), EATA (Niu et al., 2022), and SAR (Niu et al., 2023), despite constructing shift-aware graph and incorporating a single forward pass for GNN with negligible extra cost of adjusting output label estimation are required. This can be attributed to the fact that the graph we generate places each column as a node, resulting in a graph of a very small scale – typically ranging from tens to hundreds of nodes. This minimizes the cost of message passing in GNN forward process, while other baselines iterate through multiple adaptation steps with forward and backward processes, leading to increased computational expenses. Furthermore, we also provide GNN post-training time of AdapTable under different scales, encompassing small-scale (CMC, MLP), medium-scale (HELOC, FT-Transformer), and large-scale (DIABETES READMISSION, TabNet). GNN post-training requires only a few seconds for small- and medium-scale settings, and notably, it remains negligible, even in our largest experimental setting.

D.2 HYPERPARAMETER SENSITIVITY

To assess the robustness of AdapTable against hyperparameter configurations, we conduct an exhaustive hyperparameter sensitivity analysis covering all test-time parameters, including the smoothing factor α, low uncertainty quantile q_{low}, and high uncertainty quantile q_{high}. Specifically, in our experiments utilizing MLP on HELOC dataset, we perform hyperparameter optimization by varying one parameter while keeping the others fixed at the identified optimal setting, *i.e.*, $(\alpha, q_{\text{low}}, q_{\text{high}}) = (0.1, 0.25, 0.75)$. Notably, as Figure 7 exhibits, our findings reveal that the adaptation performance remains insensitive to alterations in all three types of hyperparameters, particularly when varying q_{low}, demonstrating minimal performance fluctuations. Furthermore, for the smoothing factor α and high quantile q_{high}, we pinpoint sweet spots at $[0, 0.2]$ and $[0.5, 0.6]$, respectively. This observation underscores the adaptability of our approach, allowing flexible hyperparameter selection and demonstrating generalizability across diverse test conditions. This stands in stark contrast to the hyperparameter sensitivity exhibited by the tent, as depicted in Figure 8. Notably, regardless of an extensive hyperparameter search in the tabular domain for the tent, the performance post-adaptation fails to surpass the unadapted performance, as evidenced by our main table experiment results in Table 2 and Table 3.

D.3 EFFICACY OF LABEL DISTRIBUTION HANDLER

Figure 9 analysis Jensen-Shannon Divergence value of each test batch between ground truth label distribution and prediction, comparing before and after adaptation using label distribution handler (LDH) for ANES (Studies, 2022) and DIABETES READMISSION (Clore et al., 2014) datasets, further from Section 4.3. The figure consistently exhibits a decrease in divergence after adaptation, which solidifies the efficacy of the label distribution handler 3.3.

E DATASET DETAILS

E.1 DATASETS

In our experiment, we verify our method across six different datasets. Among them, three datasets (HELOC, ANES, and DIABETES READMISSION) include natural distribution shifts between training and test data, while the other ones (CMC, MFEAT-PIXEL, and DNA) does not have such shifts, and thus we synthetically inject noises (Section E.2) on them to mimic plausible distribution shift scenarios. In our experiments, each dataset is partitioned as follows: 60% for training, 20% for validation, and 20% for testing. For all datasets, the numerical features are normalized – subtraction of mean and division by standard deviation, while categorical features are one-hot encoded. We find that different encoding types do not play a significant role in terms of accuracy, as noted in (Grinsztajn et al., 2022). Detailed specifications of each dataset are listed in Table 6

- HELOC: Home Equity Line of Credit (HELOC) (Brown et al., 2018) dataset is the dataset to predict whether the applicant will repay their HELOC account within two years; which is a line of credit typically offered by a bank as a percentage of home equity. Data is split with respect to external risk estimation value; lower ones are used for test data.
- ANES: American National Election Studies (ANES) (Studies, 2022) provide classification task of U.S. presidential election participation. Domain shift is given by the geographic region of surveyees.
- DIABETES READMISSION: Diabetes Readmission (Clore et al., 2014) represents ten years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. Each row concerns hospital records of patients diagnosed with diabetes, who underwent laboratory, medications, and stayed up to 14 days. The goal is to determine the early readmission of the patient within 30 days of discharge. Admission sources are different between train and test data.
- CMC: Contraceptive Method Choice (CMC) is a subset of Contraceptive Prevalence Survey conducted in Indonesia. The goal is to predict the current contraceptive method choice – between no-use, long-term methods, or short-term methods, with respect to the woman's demographic and socio-economic characteristics. The train and test data were split with respect to the most important column's values. Since it contains both numerical and categorical features, we obtained two different splits by selecting one most important column from the numerical columns, and one from the categorical columns.
- MFEAT-PIXEL: Multiple Features Dataset – Pixel (MFEAT-PIXEL) is a handwritten digit recognition dataset. Its goal is to classify handwritten numerals extracted from Dutch utility maps. The input is digitized, and all the pixel values are in binary form, 0 corresponding to black and 1 corresponding to white. The train and test data were split with respect to the most important column's values.
- DNA: Primate Splice-Junction Gene Sequences consist of splice junction of DNA, described by 180 indicator variables. The goal is to recognize the 3 classes – 1. boundaries between exons(retained after splicing), 2. introns(removed after splicing), or 3. none of the above. The dataset stems from Irvine database, but with major differences including the processing of symbolic variables representing the nucleotides, and the names of each example. The train and test data were split with respect to the most important column's values.

E.2 SYNTHETIC CORRUPTIONS

Let $\boldsymbol{x} = [x_1, \cdots, x_D]$ be a single table row with D columns, where μ_i and σ_i denote the mean and the standard deviation of the empirical marginal distribution of the i-th column calculated by the training set. We inject four synthetic corruptions to mimic aleatoric uncertainty, and two natural-shift oriented synthetic shifts to mimic natural distribution shifts.

- Gaussian Noise: For each column x_i, we add a Gaussian noise ϵ with $x_i \leftarrow x_i + \epsilon \cdot \sigma_i$ independently, where $\epsilon \sim \mathcal{N}(0, 0.1^2)$.
- Uniform Noise: For each column x_i, we add a uniform noise ϵ with $x_i \leftarrow x_i + \epsilon \cdot \sigma_i$ independently, where $\epsilon \sim \mathcal{U}(-0.1, 0.1)$.
- Random Missing: For each column x_i, we mask and replace it by using a random mask m_i and a random sample $\bar{x}_i$ with $x_i \leftarrow (1 - m_i) \cdot x_i + m_i \cdot \bar{x}_i$, where $m_i \sim \text{Ber}(0.2)$, $P(\bar{x}_i = k) = \sum_{i=j}^{n_s} \mathbb{1}_{[\boldsymbol{x}^s_{j,i}=k]}/n_s$ for $k \in \mathbb{R}$. n_s is the number of train instances, and $\boldsymbol{x}^s_{j,i}$ denotes the i-th column of the j-th train sample. We assume that we have knowledge of which columns are missing.

- Random Column Missing: This is similar to the random missing corruption, except for the fact that all test instances across multiple batches have the same common columns missing. For each column x_i, we mask and replace it with a random sample using a random mask m_i and a random sample $\bar{x}_i$ with $x_i \leftarrow (1 - m_i) \cdot x_i + m_i \cdot \bar{x}_i$, where $m_i \sim \text{Ber}(0.2)$, $P(\bar{x}_i = k) = \sum_{i=j}^{n_s} \mathbb{1}_{[\boldsymbol{x}^s_{j,i}=k]}/n_s$ for $k \in \mathbb{R}$. n_s is the number of train instances, and $\boldsymbol{x}^s_{j,i}$ denotes the i-th column of the j-th train sample. We also assume that we have knowledge of which columns are missing.
- Numerical Column Shift: This shift mimics natural domain shifts, we extract the most important numerical column based on pre-trained XGBoost (Chen & Guestrin, 2016) and sort all instances in the dataset according to the most important numerical column, and the top 80% of the data is predominantly allocated to the training and validation sets, while the lower 20% is primarily assigned to the test set.
- Categorical Column Shift: This shift also mimics natural domain shifts, we extract the most important categorical column based on pre-trained XGBoost (Chen & Guestrin, 2016) and split the train test dataset accordingly. Instances belonging to the category that is most frequently represented within the top 80% are predominantly assigned to the training and validation sets. Conversely, instances associated with the category that has the least frequent occurrences within the lower 20% are mainly allocated to the test set.

F Baseline Details

F.1 Supervised Baselines

- k-NN: k-Nearest Neighbors (k-NN) is a widely used model in tabular learning, that measures distance between data points using a chosen metric to identify its k-nearest neighbors, and makes predictions through majority voting for classification, or weighted averaging for regression. k is a user-defined hyperparameter, influencing the sensitivity of the model.
- LR: Logistic Regression (LR) is a linear classification algorithm for tabular data that models the probability of an instance belonging to a particular class. Using a logistic function to squash the linear combination of input features into a range of $[0, 1]$. With the appropriate regularization techniques, it has shown its capability to be comparable with SOTA architectures in the tabular domain.
- RF: Random Forest (RF) is an ensemble learning (bagging) algorithm that constructs multiple decision trees to enhance accuracy and mitigate overfitting. It excels in handling non-linear patterns, providing high accuracy and robustness against outliers.
- XGBoost: Extreme Gradient Boosting (XGBoost) (Chen & Guestrin, 2016) is an ensemble learning (boosting) algorithm building a sequence of weak learners, usually decision trees, to correct errors of the previous model. It stands out for its high predictive performance, ability to handle complex relationships and regularization features.
- CatBoost: CatBoost (Dorogush et al., 2017), similar to XGBoost, is a boosting ensemble algorithm. It efficiently handles categorical features without extensive pre-processing, making it advantageous for real-world datasets. Its benefits include high performance, but it comes at a computational cost. Additionally, parameter tuning may be necessary for optimal results.

F.2 Deep Tabular Learning Architectures

- MLP: Multi-layer perceptron (MLP) (Murtagh, 1991): is a foundational deep learning architecture characterized by multiple layers of interconnected nodes, where each node applies a non-linear activation function to a weighted sum of its inputs. In the tabular domain, MLP is often employed as a default deep learning model, with each input feature corresponding to a node in the input layer.
- TabNet: TabNet (Arik & Pfister, 2021) introduces a unique blend of decision trees and neural networks. It utilizes an attention mechanism to selectively focus on informative features at each decision step, making it particularly well-suited for handling tabular data with a mix of categorical and continuous features.

Table 7: Hyperparameter search space of supervised baselines. # Neighbors denotes the number of neighbors, # Estim denotes the number of estimators, Dep th denotes the maximum depth, and LR denotes the learning rate, respectively.

Baseline	Search Space
k-NN	# Neighbors: {2 - 12}
RF	# Estim: {50 - 200}, Depth: {2 - 12}
XGBoost	# Estim: {50 - 200}, Depth: {2 - 12}, LR: {0.01 - 1}, Gamma: {0 - 0.5}
CatBoost	# Estim: {50 - 200}, Depth: {5 - 40}

- FT-Transformer: FT-Transformer (Gorishniy et al., 2021), short for feature tokenizer along with Transformer (Vaswani et al., 2017), represents a straightforward modification of the Transformer architecture tailored for tabular data. In this model, the feature tokenizer component plays a crucial role by converting all features, whether categorical or numerical, into tokens. Subsequently, a series of Transformer layers are applied to these tokens within the Transformer component, along with the added [CLS] token. The ultimate representation of the [CLS] token in the final Transformer layer is then utilized for the prediction.

F.3 Test-Time Adaptation Baselines

- PL: Pseudo-labeling (PL) (Lee, 2013) uses a pseudo-labeling strategy to update the model weights during test-time.
- TTT++: Test-time training (TTT++) (Liu et al., 2021) tries to mitigate deterioration of test-time adaptation performance through feature alignment strategies, regularizing the adaptation, without the need to re-access source data.
- Tent: Test entropy minimization (Tent) (Wang et al., 2021a) updates the scale and bias parameters within the batch normalization layer with entropy minimization during test-time, with a given test batch.
- SAM: Sharpness-aware minimization (SAM) (Foret et al., 2021) although not a method devised for test-time adaptation, has shown its effectiveness combined with Tent through updating parameters that lie in neighborhoods having uniformly low loss.
- EATA: Efficient Anti-forgetting Test-time Adaptation (EATA) (Niu et al., 2022) points out that samples with high entropy may lead to unreliable gradients that disrupt the model. EATA filters these high-entropy samples along with utilizing a fisher regularizer to constrain important model parameters during adaptation.
- SAR: Sharpness-aware and reliable optimization (SAR) (Niu et al., 2023) improves upon SAM, armed with the observation – samples with large entropy leads to model collapse during test-time, and filters the samples for adaptation with a pre-defined threshold.
- LAME: Laplacian adjusted maximum-likelihood estimation (LAME) (Boudiaf et al., 2022) is a new approach towards test-time adaptation, adapting without parameter optimization, but only corrects the output probabilities of a classifier rather than tweaking the model's inner parameters.

G Hyperparameter Details

G.1 Supervised Baselines

For k-nearest neighbors (k-NN), logistic regression (LR), random forest (RF), XGBoost (Chen & Guestrin, 2016), and CatBoost (Dorogush et al., 2017), optimal parameters are searched for each datasets using random search of 100 iterations, for each dataset. The search space for each method is specified in Table 7.

Table 8: Hyperparameter search space of test-time adaptation baselines. Here, we only denote the common hyperparameters, where method specific hyperparameters are specified in Section G.2.

Hyperparameter	Search Space
Learning Rate	{1e-3, 1e-4, 1e-5, 1e-6}
Adaptation Steps	{1, 5, 10, 15, 20}
Episodic	{True, False}

Table 9: Selected hyperparameters of test-time adaptation baselines. In this table we only denote the common hyperparameters, where method specific hyperparameters are specified in text.

Baseline	Learning Rate	Adaptation Steps	Episodic
PL	1e-4	1	True
TTT++	1e-5	10	True
TENT	1e-4	1	True
SAM	1e-3	1	True
EATA	1e-5	10	True
SAR	1e-3	1	True
LAME	N/A	N/A	N/A

G.2 Test-Time Adaptation Baselines

Entropy minimization-based methods, namely TENT Wang et al. (2021a), SAM Foret et al. (2021), and SAR Niu et al. (2023), require 2 main hyperparameters – learning rate, number of adaptation steps per batch, and whether to reset the model after batch (*i.e.*, episodic adaptation). Additionally, SAR Niu et al. (2023) requires a threshold hyperparameter to filter samples with high entropy. For TENT, we set the learning rate as 0.0001 with 1 adaptation step and episodic update. For SAM Foret et al. (2021) and SAR Niu et al. (2023), the learning rate is 0.001 with 1 adaptation step and episodic update. For PL Lee (2013), we set the learning rate as 0.0001 with 1 adaptation step and episodic updates. For TTT++ (Liu et al., 2021), EATA (Niu et al., 2022) and LAME (Boudiaf et al., 2022), we find that the author's hyperparameter choices are optimal, as specified in their paper and official code, except for their learning rate and adaptation steps. For TTT++ (Liu et al., 2021) and EATA (Niu et al., 2022), the learning rate is set to 0.00001 with 10 adaptation steps per batch and episodic updates. LAME Boudiaf et al. (2022) only corrects the output logits, thus not requiring hyperparameters related to gradient updates. For all previous baselines, we find that their hyperparameter choice did not vary across different architectures, namely MLP, TabNet (Arik & Pfister, 2021) and FT-Transformer (Gorishniy et al., 2021). As noted in the main paper, all hyperparameters for the corresponding method and backbone architecture pair are tuned with respect to numerical shift on CMC dataset from OpenML-CC18 (Bischl et al., 2021). An overview of the hyperparameter search space, along with selected hyperparameters of each method is provided in Table 8 and Table 9, respectively.

G.3 AdapTable

AdapTable requires three important hyperparameters: smoothing factor α, and low/high uncertainty quantiles $q_{\text{low}}/q_{\text{high}}$. The parameters for each backbone architecture are described in Table 10.

H Limitations and Broader Impacts

H.1 Limitations

Similar to other test-time training (TTT) methods, AdapTable incorporates an additional training procedure during the source model's training phase. This contrasts with fully test-time adaptation methods, which refrain from making assumptions during test-time execution. Specifically, AdapTable necessitates an extra post-training step for shift-aware uncertainty calibrator to adjust the model's predictions. While fully test-time adaptation methods, such as SAR (Niu et al., 2023), are applicable,

Table 10: Selected hyperparameters of AdapTable. Three major hyperparameters – smoothing factor α, low quantile q_{low}, high quantile q_{high} are specified below per architecture. The hyperparameters were fixed throughout datasets.

Architecture	α	q_{low}	q_{high}
MLP	0.1	0.25	0.75
TabNet	0.0	0.25	0.9
FT-Transformer	0.0	0.25	0.9

their performance improvements in the tabular domain are limited, often failing to address certain covariate/label shifts, as evidenced in our evaluations. AdapTable demonstrates substantial performance gains in the majority of evaluation scenarios, although occasional shortcomings persist in specific datasets and model specifications.

H.2 Broader Impacts

Our research represents an initial effort to address the challenge of domain shift, a significant impediment in the practical deployment of machine learning models for tabular data. Despite the prevalence of tabular data in industrial applications, investigations into domain adaptation specific to this data type have been comparatively limited, especially when contrasted with domains such as computer vision, natural language processing, and speech processing.

Through comprehensive examinations, we have identified that the straightforward application of TTA methodologies from other domains, particularly those relying on entropy minimization, which currently constitutes the most prevalent form of TTA, encounters substantial challenges in the context of tabular data. Notably, these challenges arise from the high uncertainty of prediction entropy for tabular data, potentially leading to model collapse, as exemplified in SAR (Niu et al., 2023), and the inadequacy of the cluster assumption within the latent space of models trained on tabular data. Moreover, most TTA methodologies are tailored exclusively to deep learning models, an assumption often overlooked in domains where deep learning models have surpassed traditional machine learning approaches. However, this assumption cannot be dismissed in the tabular domain, where classical machine learning methods, such as decision trees, form a competitive baseline.

In contrast, AdapTable, by refining only the output probabilities and circumventing noisy backpropagation from high-entropy-prone data, avoids failure. Additionally, our novel shift-aware uncertainty calibrator leverages the heterogeneous characteristics of columns, enabling our method to effectively address domain shift. We posit that our work empowers future researchers to adeptly confront the crucial challenge of mitigating domain shift in tabular data – an arena where the application of prior methods from other domains is not straightforward due to the aforementioned issues.

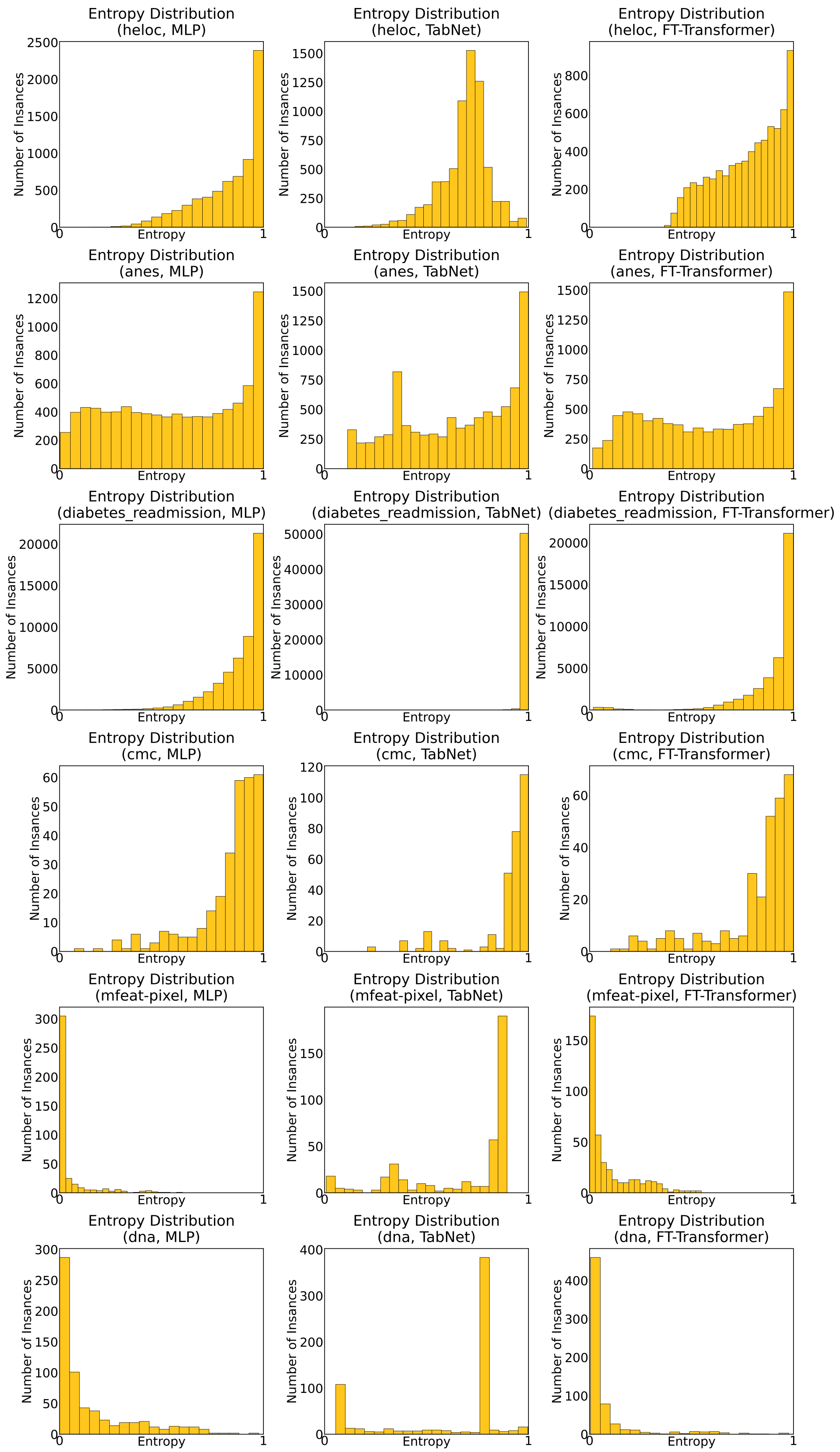

Figure 10: Entropy distribution histograms for test instances across six different datasets and three representative deep tabular learning architectures.

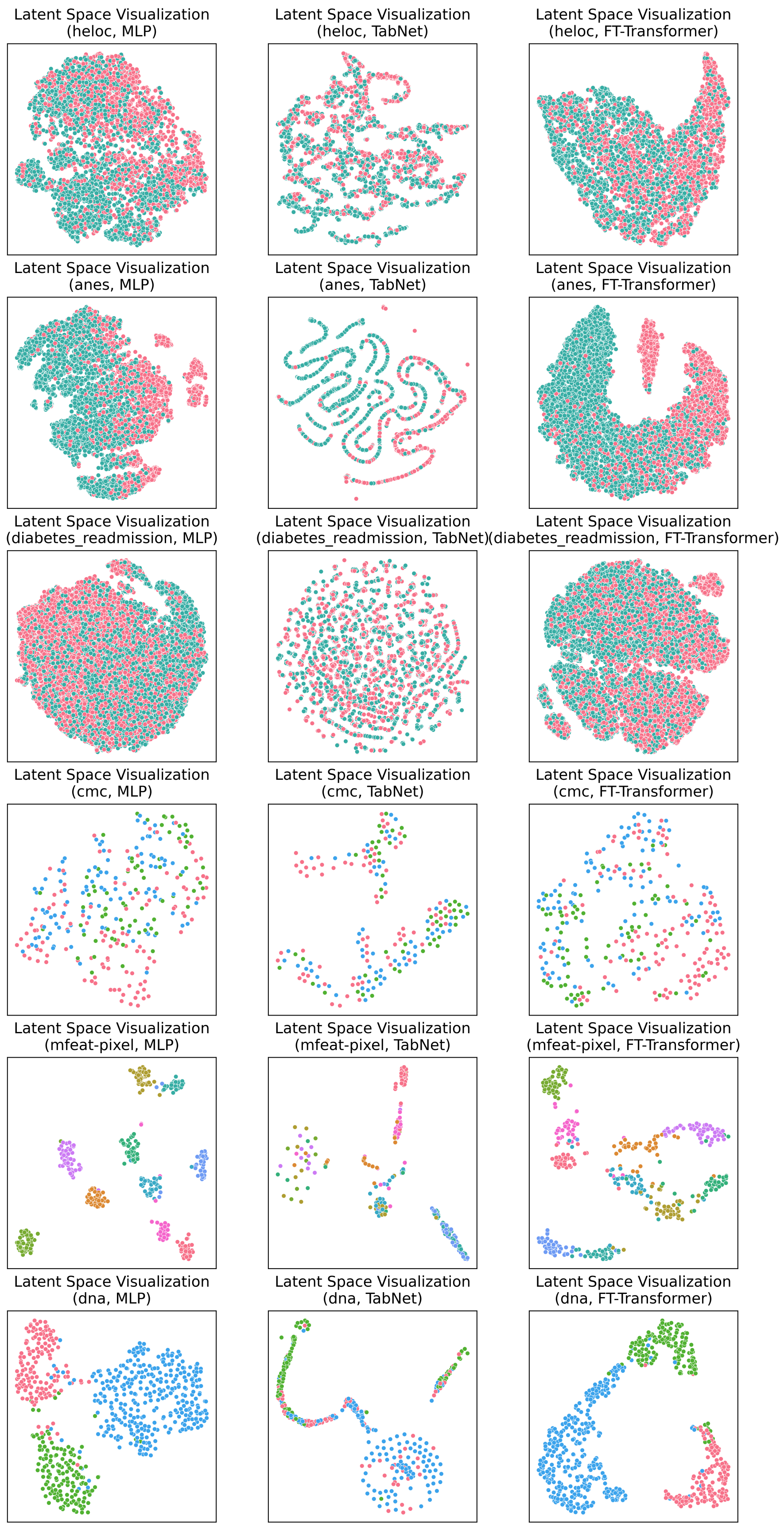

Figure 11: Latent space visualizations for test instances using t-SNE across six different datasets and three representative deep tabular learning architectures.

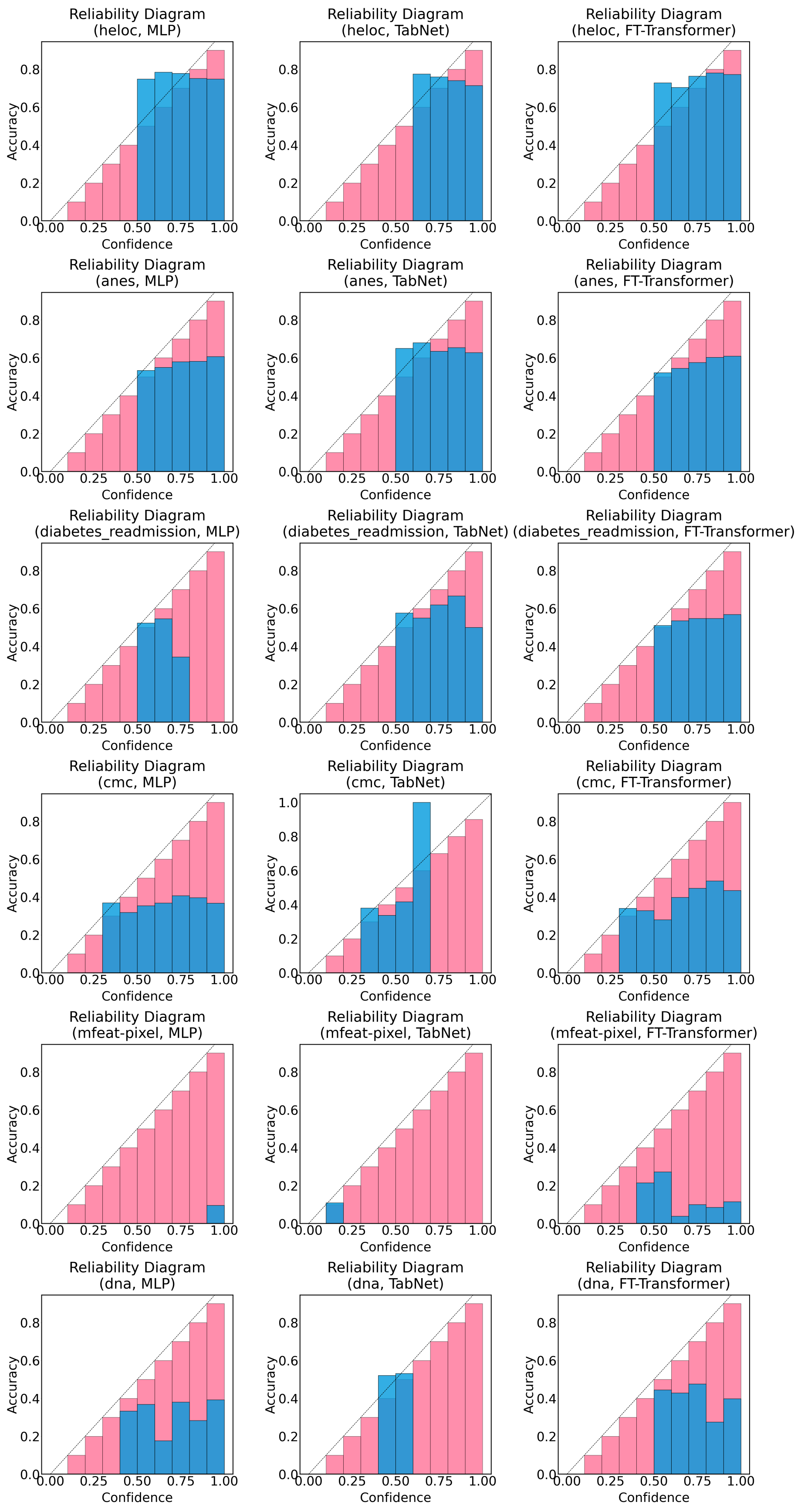

Figure 12: Reliability diagrams for test instances across six different datasets and three representative deep tabular learning architectures.